Sommatore a 3 input (v2)

Stefano Scarcelli & Michele De Fusco

27 Dic 2023

Contents

1 Analisi progettuale

1.1 Analisi preliminare

L'obbiettivo è quello di costruire un circuito in grado di sommare 3 numeri a *n-bit* (*2's complements*) e restituirne il risultato. Sia gli input che gli output devono essere sincronizzati tramite l'uso di registri.

1.2 Struttura del progetto

La struttura del progetto sarà fatta da un modulo che implementa i **registri** e un'altro dove viene implementato il circuito finale (**Three adder**).

2 Implementazione

2.1 FullAdder

L'implementazione dell'**FullAdder** segue l'implementazione classica.

2.1.1 Codice VHDL

```
----------------------------------------------------------------------------------
-- Company: UNICAL
-- Engineer: Michele De Fusco
--
-- Create Date: 29.12.2023 13:05:14
-- Design Name: ---
-- Module Name: FullAdder - Behavioral
-- Project Name: ThreeNumbersAdder
-- Target Devices: xc7z020clg400-2
-- Tool Versions: 2023.2
-- Description: Full adder
--
-- Dependencies: None
--
-- Revision: 1
-- Revision 1.0 - Implementation
-- Additional Comments: ---
--
----------------------------------------------------------------------------------
```

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity FullAdder is
    Port ( a : in STD_LOGIC;
           b : in STD_LOGIC;
           c : in STD_LOGIC;
           cout : out STD_LOGIC;
           r : out STD_LOGIC);
end FullAdder;

architecture Behavioral of FullAdder is
    signal p : STD_LOGIC;
begin
    p <= a xor b;
    r <= p xor c;
    cout <= (a nand b) nand (c nand p);
end Behavioral;
```

2.1.2 Sintesi

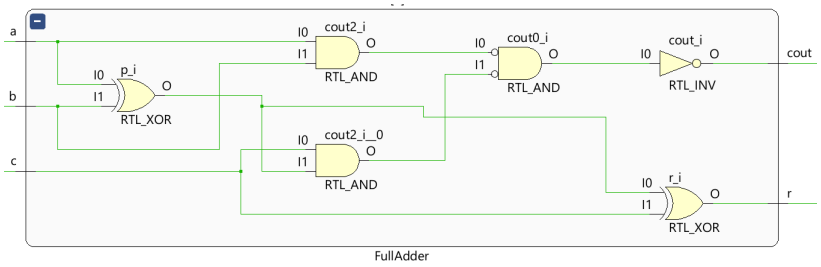

Figure 1: **FullAdder**.

2.2 Adder n-bit

L'implementazione dell'**Adder n-bit** è la medesima del progetto precedente (v1).

2.2.1 Codice VHDL

```
-- Company: UNICAL
-- Engineer: Stefano Scarcelli, Michele De Fusco
--
-- Create Date: 29.12.2023 12:31:56
-- Design Name:
-- Module Name: Adder - Behavioral
-- Project Name: ThreeNumbersAdder
-- Target Devices: xc7z020clg400-2
-- Tool Versions: 2023.2
-- Description: Adder n-bit
--
-- Dependencies: None
--
-- Revision: 2
-- Revision 1.0 - Implementation
-- Additional Comments: ---
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Adder is
    generic (n: integer := 8);
    Port ( A : in STD_LOGIC_VECTOR (n-1 downto 0);
           B : in STD_LOGIC_VECTOR (n-1 downto 0);
           R : out STD_LOGIC_VECTOR (n downto 0));
end Adder;

architecture Behavioral of Adder is

signal A1, B1, P, G: STD_LOGIC_VECTOR (n downto 0);
signal C: STD_LOGIC_VECTOR (n+1 downto 0);

begin
    P <= A1 xor B1;
    G <= A1 and B1;
    A1 <= A(n-1)&A;
    B1 <= B(n-1)&B;
    C(0) <= '0';
    C(n+1 downto 1) <= G or (P and C(n downto 0));
    R <= P xor C(n downto 0);
end Behavioral;
```

2.2.2 Sintesi

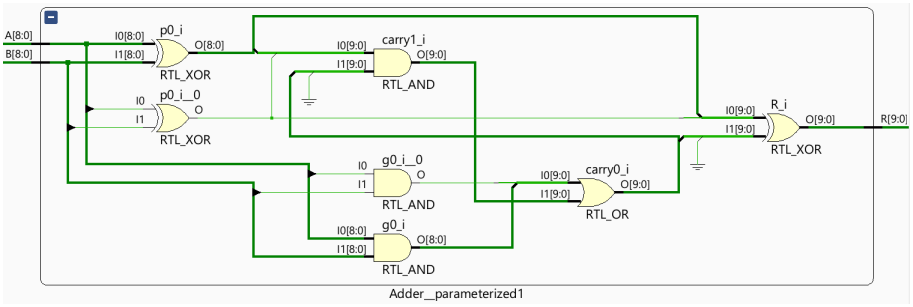

Figure 2: **FullAdder**.

2.3 Register n-bit

L'implementazione dell'**Register n-bit** è la medesima del progetto precedente (v1).

2.3.1 Codice VHDL

```
----------------------------------------------------------------------------------
-- Company: UNICAL
-- Engineer: Stefano Scarcelli
--
-- Create Date: 04.12.2023 14:57:11
-- Design Name: ---
-- Module Name: Register_n - Version1
-- Project Name: ThreeNumbersAdder
-- Target Devices: xc7z020clg400-2
-- Tool Versions: 2023.2
-- Description: Parametric n-bit register
--
-- Dependencies: None
--
-- Revision: 1
-- Revision 1.0 - Implementation
-- Additional Comments: ---
--
----------------------------------------------------------------------------------
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Register_n is
    generic (n : integer := 8);
    Port (CLK, Clear : in STD_LOGIC;
          D : in STD_LOGIC_VECTOR (n-1 downto 0);
          Q : out STD_LOGIC_VECTOR (n-1 downto 0));
end Register_n;

architecture Version1 of Register_n is

begin
    process(CLK, Clear) begin
        if (Clear = '1') then
            Q <= (others => '0');
        elsif rising_edge(CLK) then
            Q <= D;
        end if;
    end process;

end Version1;
```

2.3.2 Sintesi

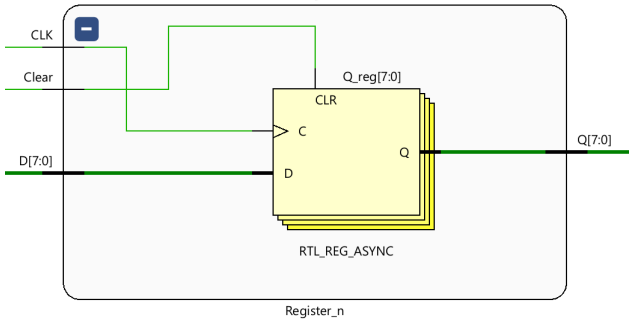

Figure 3: **Registro** a 8bit.

2.4 Three adder (main)

L'implementazione del modulo principale assembla i vari componenti precedentemente definiti in modo adeguato per eseguire la somma.

2.4.1 Codice VHDL

```
----------------------------------------------------------------------------------
-- Company: UNICAL
-- Engineer: Stefano Scarcelli, Michele De Fusco
--
-- Create Date: 26.12.2023 17:10:58
-- Design Name: ---
-- Module Name: Three_Adder - Version2
-- Project Name: ThreeNumbersAdder
-- Target Devices: xc7z020clg400-2
-- Tool Versions: 2023.2
-- Description: Main file of the project
--
-- Dependencies: Register_n.vhd
--
-- Revision: 3
-- Revision 1.0 - Implementation
-- Additional Comments: ---
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Three_Adder is
    generic (n : integer := 8);
    Port (A : in STD_LOGIC_VECTOR (n-1 downto 0);
          B : in STD_LOGIC_VECTOR (n-1 downto 0);
          C : in STD_LOGIC_VECTOR (n-1 downto 0);
          R : out STD_LOGIC_VECTOR (n+1 downto 0);
          CLK : in STD_LOGIC;
          Clear : in STD_LOGIC);
end Three_Adder;

architecture Version2 of Three_Adder is
    component Register_n
        generic (n : integer := 8);
        Port (CLK, Clear : in STD_LOGIC;
```

```
            D : in STD_LOGIC_VECTOR (n-1 downto 0);
            Q : out STD_LOGIC_VECTOR (n-1 downto 0));
    end component;

    component FullAdder
    Port ( a : in STD_LOGIC;
           b : in STD_LOGIC;
           c : in STD_LOGIC;
           cout : out STD_LOGIC;
           r : out STD_LOGIC);
    end component;

    component Adder is
    generic (n: integer := 8);
    Port ( A : in STD_LOGIC_VECTOR (n-1 downto 0);
           B : in STD_LOGIC_VECTOR (n-1 downto 0);
           R : out STD_LOGIC_VECTOR (n downto 0));
    end component;

    signal Ra, Rb, Rc: STD_LOGIC_VECTOR (n-1 downto 0);
    signal Rs: STD_LOGIC_VECTOR (n+1 downto 0);

    signal vr, sp: STD_LOGIC_VECTOR(n downto 0);
    signal vr1, sp1: STD_LOGIC_VECTOR(n-1 downto 0);
begin
    RegA: Register_n generic map(n) port map(CLK, Clear, A, Ra);
    RegB: Register_n generic map(n) port map(CLK, Clear, B, Rb);
    RegC: Register_n generic map(n) port map(CLK, Clear, C, Rc);

    s1: for i in 0 to (n-1) generate
        FAi: FullAdder port map(Ra(i), Rb(i), Rc(i), vr1(i), sp1(i));
    end generate;

    sp <= sp1(n-1)&sp1;
    vr <= vr1&'0';

    Sum: Adder generic map(n+1) port map(vr, sp, Rs);

    RegS: Register_n generic map(n+2) port map(CLK, Clear, Rs, R);
end Version2;
```

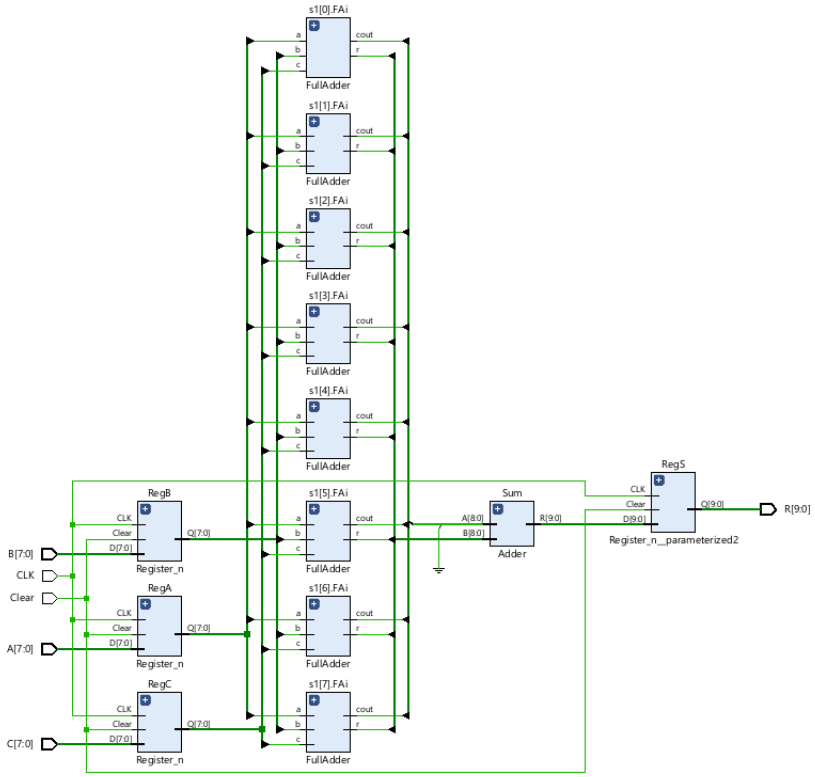

Figure 4: Circuito principale.

2.4.2 Sintesi

2.5 Time constraint

In più abbiamo aggiunto un *time constraint* sul segnale di clock con un periodo di **5ns**.

```
create_clock -period 5.000 -name main_clock
-waveform {0.000 2.500} [get_nets CLK]
```

2.5.1 Analisi del Time constraint

Con l'implementazione del codice abbiamo verificato (*Timing summary routed*) che il **Time constraint** fosse valido, valutando di aver inserito un periodo di clock di superiore rispetto al minimo tollerabile dal circuito di **2.219ns**, rispetto dai **1.958ns** del progetto precedente (v1).

2.6 Risorse

Name	Slice LUTs (53200)	Slice Registers (106400)	Slice (13300)	LUT as Logic (53200)	Bonded IOB (125)	BUFGCTRL (32)
N Three_Adder	24	34	16	24	36	1
RegA (Register_n)	0	8	7	0	0	0
RegB (Register_n_0)	14	8	9	14	0	0
RegC (Register_n_1)	13	8	9	13	0	0
RegS (Register_n__parameterized2)	0	10	6	0	0	0

Figure 5: Risorse usate dai vari componenti.

Resource	Utilization	Available	Utilization %
LUT	24	53200	0.05
FF	34	106400	0.03
IO	36	125	28.80

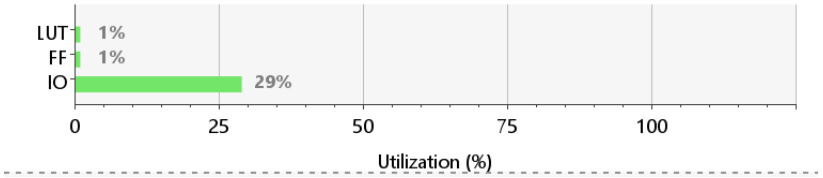

Figure 6: Riassunto delle risorse utilizzate.

2.6.1 Potenza

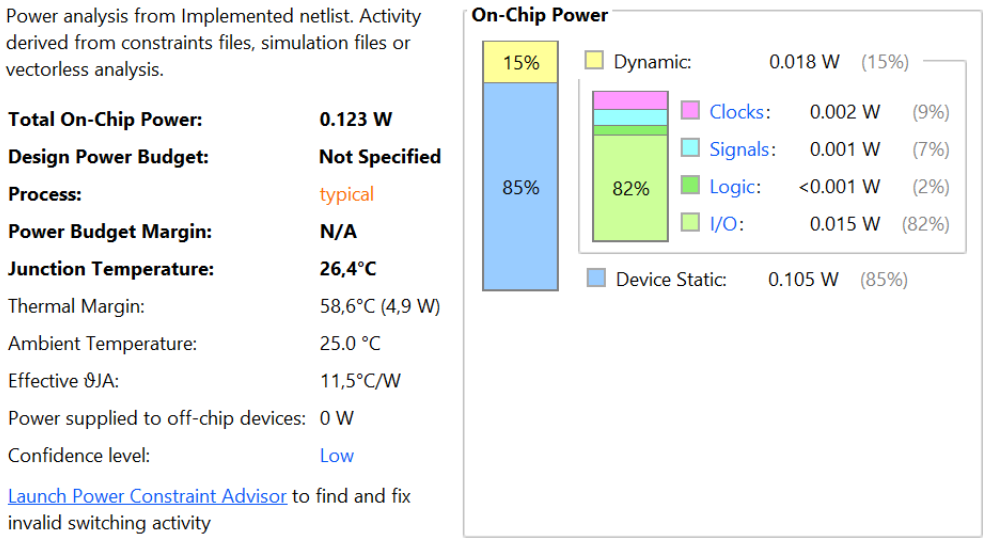

Figure 7: Riassunto del report della potenza del circuito.

3 Testing

Per il test abbiamo deciso di usare il medesimo test del progetto precedente (v1).

3.1 Codice test test-bench

```
-- Company:
-- Engineer: Stefano Scarcelli, Michele De Fusco
--
-- Create Date: 12.12.2023 11:20:24
-- Design Name: ----
-- Module Name: Sim_Add - Version1
-- Project Name: ThreeNumbersAdder
-- Target Devices: xc7z020clg400-2
-- Tool Versions: 2023.2
-- Description: Main simulation
```

```
--
-- Dependencies: Three_Adder.vhd
--
-- Revision: 1
-- Revision 2.0 - Implementation
-- Additional Comments: ---
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_arith.ALL;

entity Sim_Add is
    generic (n : integer := 8);
end Sim_Add;

architecture Version1 of Sim_Add is
    component Three_Adder
        -- generic (n : integer := 8);
        Port (A : in STD_LOGIC_VECTOR (n-1 downto 0);
              B : in STD_LOGIC_VECTOR (n-1 downto 0);
              C : in STD_LOGIC_VECTOR (n-1 downto 0);
              R : out STD_LOGIC_VECTOR (n+1 downto 0);
              CLK : in STD_LOGIC;
              Clear : in STD_LOGIC);
    end component;
    Signal IA, IB, IC : STD_LOGIC_VECTOR (n-1 downto 0);
    Signal ORR : STD_LOGIC_VECTOR (n+1 downto 0);
    signal clk, clear : STD_LOGIC;
    constant T : time := 10 ns;
begin
    TA : Three_Adder port map(IA, IB, IC, ORR, clk, clear);
    process begin
        clk <= '0';
        wait for T/2;
        clk <= '1';
        wait for T/2;
    end process;

    process begin
        wait for 100 ns;

        clear <= '1';
```

```
        wait for T;
        clear <= '0';
        wait for T/4;

        -- (-1)+(-1)+(-1)=-3
        IA <= (others=>'1');
        IB <= (others=>'1');
        IC <= (others=>'1');
        wait for T;

        -- 127+127+127=381
        IA(n-1) <= ('0');
        IA(n-2 downto 0) <= (others=>'1');
        IB(n-1) <= ('0');
        IB(n-2 downto 0) <= (others=>'1');
        IC(n-1) <= ('0');
        IC(n-2 downto 0) <= (others=>'1');
        wait for T;

        -- (-64)+1+(-63)=0
        IA(n-1 downto n-2) <= (others=>'0');
        IA(n-3 downto 0) <= (others=>'1');
        IB(n-1 downto 1) <= (others=>'0');
        IB(0) <= ('1');
        IC(n-1 downto n-2) <= (others=>'1');
        IC(n-3 downto 0) <= (others=>'0');
        wait for T;

        -- Waiting results
        wait for T;
    end process;

end Version1;
```

3.2 Risultati

I risultati delle simulazioni confermano il funzionamento del circuito come previsto.

3.2.1 Simulazione comportamentale

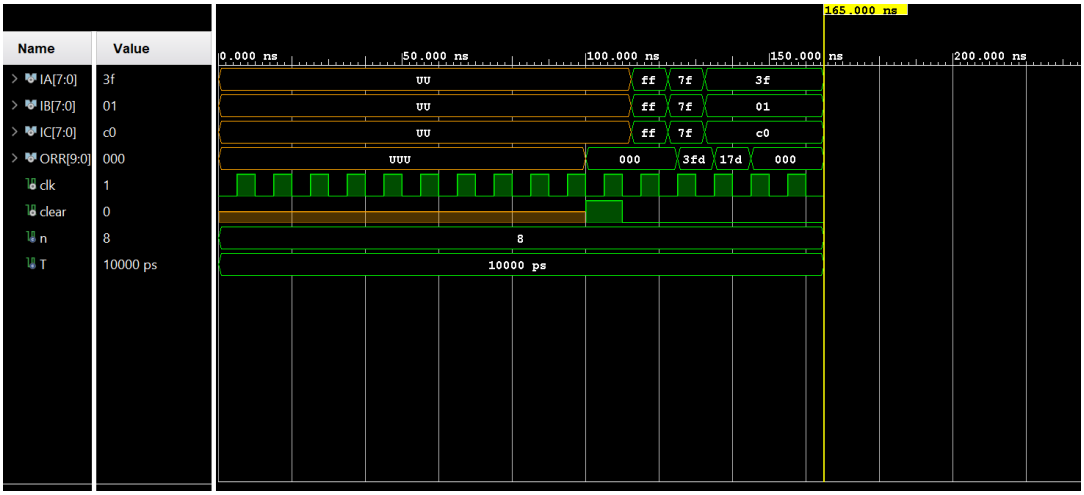

Figure 8: Grafico temporale simulazione comportamentale.

3.2.2 Simulazione post sintesi

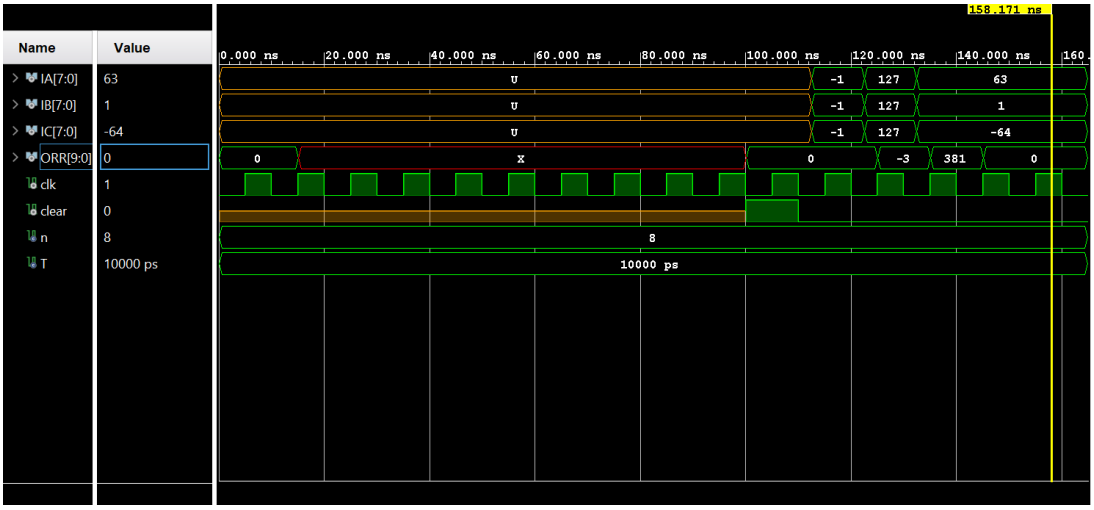

Figure 9: Grafico temporale simulazione post sintesi.

3.2.3 Simulazione post implementazione

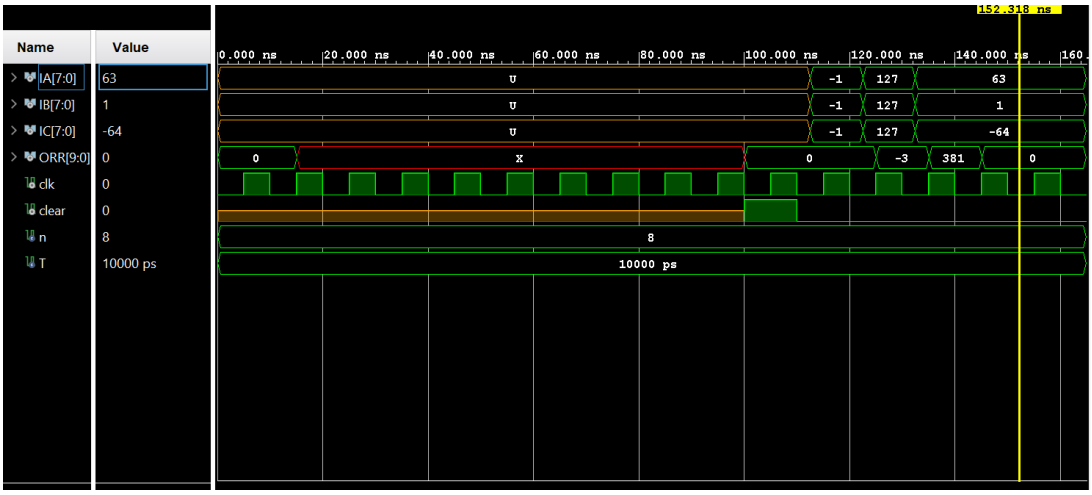

Figure 10: Grafico temporale simulazione post implementazione.